上篇

基礎篇

ARTICLE

第1章 C语言上机实践的目的与要求

上机实践是学习程序设计语言时必不可少的一个实践环节，特别是C语言灵活、简洁，更需要通过实际的编程实践来真正掌握它。程序设计语言的学习目的可以概括为学习语法规则、掌握程序设计方法和提高程序开发能力这几方面，而这些都必须通过充分的上机实践操作才能完成，在明确上机实践的目的与要求前，首先必须了解C程序的开发步骤。

1.1 C程序的开发步骤

用高级语言编写的程序被称为源文件，需要将其转换为二进制机器代码才能在计算机上运行。转换过程分为编译和链接两步进行，首先需要对源文件进行编译，生成的文件被称为目标文件，然后将目标文件进行链接，生成的文件被称为可执行文件。可执行文件才可以在计算机上运行。C程序的开发步骤如图1.1所示。

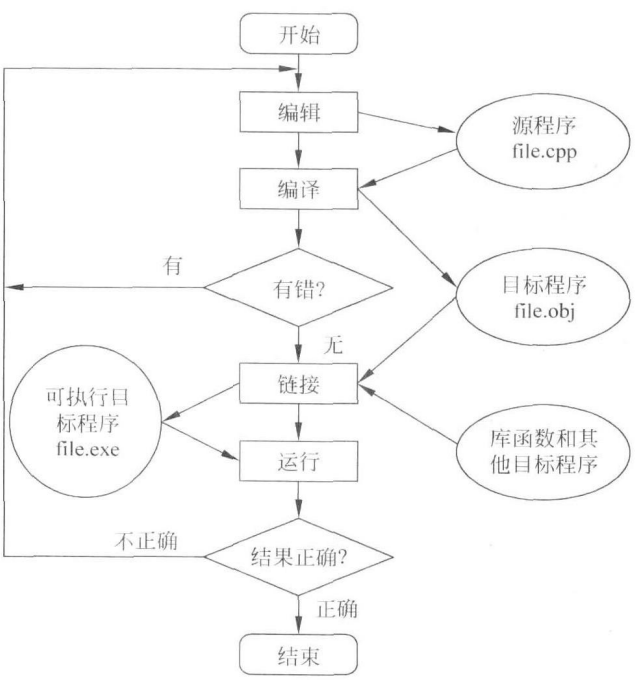


图1.1 C程序的开发步骤

1.2 C程序上机实践的目的

从C程序的开发步骤读者可以知道“C程序设计”是一门实践性很强的学科，因此，在学习“C程序设计”课程时就不能满足于能看懂书上的程序，而应当熟练地掌握程序设计的全过程，充分重视上机实践环节，独立编写源程序，独立上机调试程序，独立运行程序和分析结果。

但是，上机实践的目的，绝不仅仅是为了验证教材和讲课的内容或者验证自己所编的程序正确与否。学习程序设计时，上机实践的目的包括以下几方面。

1. 加深对课堂讲授内容的理解

C语言有许多语法规定，光靠课堂讲授，难免让人感觉枯燥乏味，难以记住这些规定，但它们又都是编写程序时很重要的规则。而通过实际的上机操作，就能自然、熟练地掌握这些规则。因此通过上机来掌握语法规则是行之有效的方法。

2. 熟悉程序开发环境、学习计算机系统的操作方法

程序必须在一定的外部环境下才能运行，所谓“环境”，就是指所使用的计算机系统的硬件和软件条件，只有学会利用这些环境，才能顺利地进行程序开发工作。而通过上机实践，就能熟练掌握C语言开发环境，为以后编写计算机程序解决实际问题打下基础。同时，还能在遇到其他开发环境时触类旁通，迅速地掌握利用新系统进行开发工作的方法。本书主要介绍了Turbo C++ 3.0的运行环境，以及Visual C++ 6.0运行环境下C程序的设计与调试方法。

3. 学会上机调试程序

上机调试程序就是发现程序中的错误，并且能迅速地排除这些错误，使程序正常运行。经常上机的人遇到的问题多，而处理问题时积累的经验也丰富，因此当编译出现“出错信息”时，就能很快地判断出错误所在，并迅速解决问题。而这样的经验和能力只有通过长期上机实践才能获得。学习别人调试程序的经验固然重要，但更重要的是自己能通过上机，体会、分析和总结编写、调试程序时的经验和心得，而且有些经验是只能“意会”，难以“言传”的。

计算机技术是一门实践性很强的技术，要求从事这一工作的人不仅能了解和熟悉相关理论和方法，还要求具备动手实践的能力。对程序设计员来说，编写程序并上机调试通过是最基本的技能要求，因此调试程序本身是程序设计课程的一个非常重要的内容，也是一个基本要求，同学们应给予充分的重视。在上机实践时千万不要在程序通过后就认为万事大吉、完成任务了，而应当在已经通过的程序基础上做一些改动（例如修改一些参数、增加程序的一些功能、改变输入数据的方法等），再进行编译、链接和运行。在按照本教材上机实践时，应认真完成每个上机实践内容及有关思考题部分。在程序调试完毕后，还可以“自设障碍”，即把正确的程序改为有错的（例如用scanf函数输入变量时，故意漏写“&”符号、使数组下标出界、使整数溢出等），以便于观察和分析所出现的情况，掌握各种调试技能。这种学习就是一种灵活、主动的学习方法，而不是呆板被动的学习，这样才会有真正收获。

1.3 C 程序上机实践的步骤与要求

学习 C 语言，算法分析与设计是核心，而新颖的任务能促使学生主动地探寻问题的算法。因此，针对上机实践环节，专门编写了本书，在编写本书时，以培养学生阅读程序、编写程序的能力，激发学生编程热情为目标，精心组织每个上机实验内容，结合当前计算机等级考试上机考试题型，每个上机实验的实践内容分为程序填空、程序改错、程序设计三种题型，每个题目又有上机实践提示及关联思考，要求学生在 1～1.5 小时内完成。

学生要想在有限的时间内完成每一个上机实验，上机前必须自主学习，做好充分的预习准备工作，写好预习报告，明确本次上机实践的目的与要求，复习与本次上机实践有关的教学内容，将程序填空、程序改错、程序设计全部做好，写出每个题目的算法分析及 N-S 结构流程图，同时模拟计算机运行过程将程序的运行结果写出来，然后在上机实践时调试程序，比较两次结果有何不同。完成本次上机实践后要针对每个题目进行总结，分析在调试过程中遇到的问题及解决方法，确认本次上机实践是否达到预期上机实践目的。

因此，上机实践一般包括预习、上机调试和实践总结三个步骤，具体如下。

1. 准备好上机所需的程序和测试数据

根据问题，进行分析，选择适当的算法并编写程序。上机前一定要仔细检查程序（称为静态检查），直到找不到错误（包括语法错误和逻辑错误）。然后分析和考虑可能会遇到的问题及解决的对策，并准备几组测试程序的数据和预期结果，以便发现程序中可能存在的错误，提高上机效率。对程序中有疑问的地方，应做出记号，在上机时给予更多的关注。初学者切忌抄袭他人的程序去上机，应从一开始就养成严谨的科学作风。

2. 上机输入、编辑和调试程序

应该一人一组独立上机。在调试过程中，应充分地学习和使用 C 语言集成开发环境提供的各种调试手段和工具。使用预先准备好的测试数据运行程序，观察是否得到预期的正确结果。对于上机过程中出现的问题，除了因系统而引起的问题以外，一般应独立处理，不要轻易举手问老师。尤其在出现“出错信息”时，应善于独立进行分析判断，因为这是学习调试程序的最好机会。通过实践得到的经验才是记忆最深刻、掌握最牢固的知识。

在使用键盘时，还可以利用以前所学的英文指法知识，练习以正确的指法击键输入。

3. 上机结束后，整理上机实践结果，认真分析和总结，写实验报告

对上机实践结果认真分析，判断是否达到了题目要求；对上机实践内容认真归纳总结，评估是否达到了本次上机实践的目的与要求；对上机实践过程中遇到的问题也应分析和总结，以避免以后犯相同的错误。最后还要根据教师要求写出上机实践报告。

上机实践报告一般应包括以下内容：

（1）上机实践内容：包含上机实践题目与要求。

（2）算法说明：用文字或流程图说明。

（3）程序清单。

（4）运行结果：包含原始数据、相应的运行结果和必要的说明。

（5）分析与思考：调试过程中遇到的问题及解决办法；调试程序的心得与体会；其他算法的存在与实践等。若最终未完成调试，要认真找出错误并分析原因。

第2章 Turbo C++ 3.0环境下的C程序设计与调试

Turbo C++ 3.0 是一个集程序编辑、编译、链接和调试为一体的 C 语言程序开发软件，具有速度快、效率高、功能强等优点，使用非常方便。可在 Turbo C++ 3.0 环境下进行全屏幕编辑，利用窗口功能进行编译、链接、调试、运行和环境设置等工作。

2.1 Turbo C++ 3.0 系统的建立

Turbo C++ 3.0 是基于 DOS 平台的编译系统，要求计算机安装 DOS 2.0 以上版本，它占用硬盘不到 3.5MB，使用时占用内存 384KB，对鼠标、显示器无特殊要求，几乎目前所有计算机都可以使用它。Turbo C++ 3.0 系统一般以压缩文件形式提供，首先将其解压复制到本计算机上任一目录。一般将 TC 环境解压复制到 C 盘的 TC 目录下，在该目录下有下列子目录和文件：

(1) C:\TC\BIN 子目录：其中包括 tc.exe、tcc.exe、make.exe、tlink.exe、tlib.exe 等可执行文件。

(2) C:\TC\INCLUDE 子目录：其中包括 stdio.h、math.h、malloc.h、string.h 等头文件。

(3) C:\TC\LIB 子目录：其中包括 maths.lib、mathl.lib、graphics.lib 等库函数文件。

以后如无特别声明，假定 TC 环境安装在 C 盘上 TC 子目录下。

2.2 Turbo C++ 3.0 集成环境的使用

2.2.1 进入 Turbo C++ 3.0

进入 Turbo C++ 3.0 有两种方式。

(1) 从资源管理器进入 C:\TC\BIN 目录，双击 TC.exe 可执行文件，就会出现 Turbo C++ 3.0 集成环境。

(2) Turbo C++ 3.0 是学习中要经常使用的一个应用程序，因此机房管理员通常会把 Turbo C++ 3.0 的快捷方式放到 Windows 桌面上(如图 2.1 所示)，因此在 Windows 桌面上，双击 Turbo C++ 3.0 的快捷方式图标也可以快速进入 Turbo C++ 3.0 集成环境。

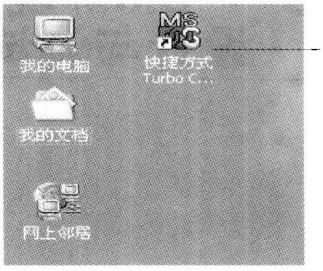


图 2.1 Turbo C++ 3.0 快捷方式图标

进入 Turbo C++ 3.0 集成开发环境后，其运行界面如图 2.2 所示。

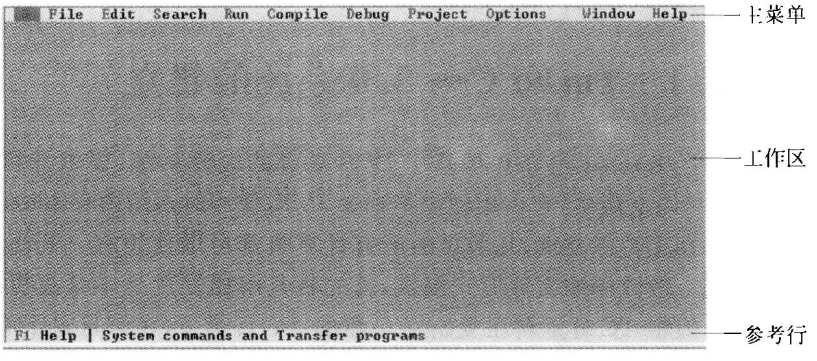


图 2.2 Turbo C++ 3.0 主界面

2.2.2 选择工作目录

工作目录指的是用户文件所在的目录。为方便查找所编写的文件，用户最好建立一个自己专用的子目录，即文件夹。用户可以在此文件夹下进行文件的编辑工作，编译生成的目标文件也可以存放在此文件夹中。假如 030111 班的一位同学已在 D 盘下创建了一个自己专用的“D:\030111”文件夹，然后再用“Change dir”命令将 Turbo C++ 3.0 工作目录修改为自己所设定的工作目录，操作方法如下。

(1) 选中 File 后，按回车键，File 菜单将弹出一个子菜单，如图 2.3 所示。

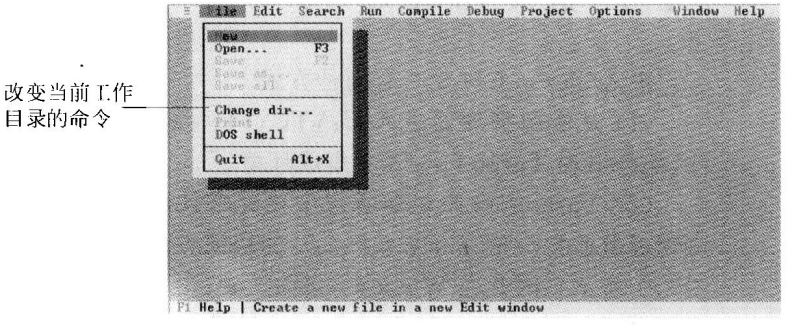


图 2.3 改变当前工作目录的命令

(2) 用上下光标移动键“↑”、“↓”键，选中 Change dir 命令，或者直接按 C 键来选中 Change dir 命令。随后按回车键，将弹出如图 2.4 所示的对话框，提示用户输入所选择的用户工作目录。

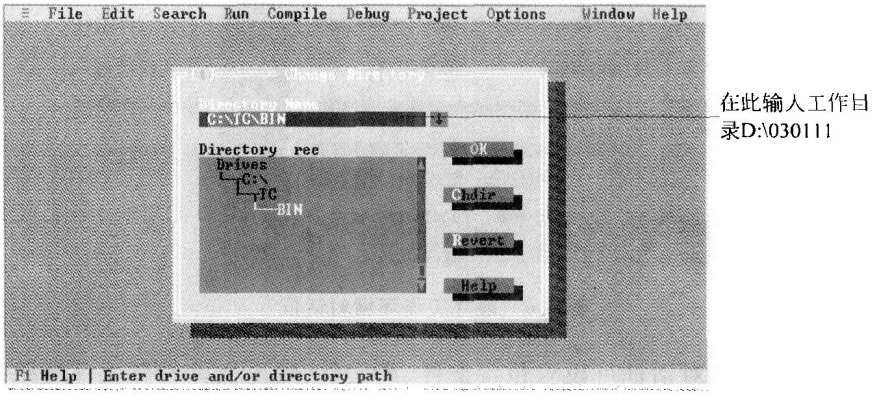


图 2.4 改变当前工作目录

(3) 用户可以把 Directory Name 文本框中默认的工作目录“C:\TC\BIN”改为用户的工作目录，如“D:\030111”，随后按回车键或者单击【OK】按钮即可。

注意：此工作目录必须事先已经创建，否则将会显示“此路径无效”的错误。

(4) 用户还可以在 Directory Tree 下的目录树结构中双击鼠标左键，选择已经创建好了的用户工作目录所在路径，用此方法也可以达到改变工作目录的目的。

2.2.3 建立工作环境

建立工作环境也就是告诉 Turbo C++：包含文件和库文件在什么地方；输出文件存于何处。可以使用 Options 菜单来进行这项设置，操作步骤如下。

(1) 在主菜单窗口中选中 Options 菜单，按回车键。此时 Options 菜单下将弹出子菜单，用光标上下移动键选中 Directories 命令，如图 2.5 所示。

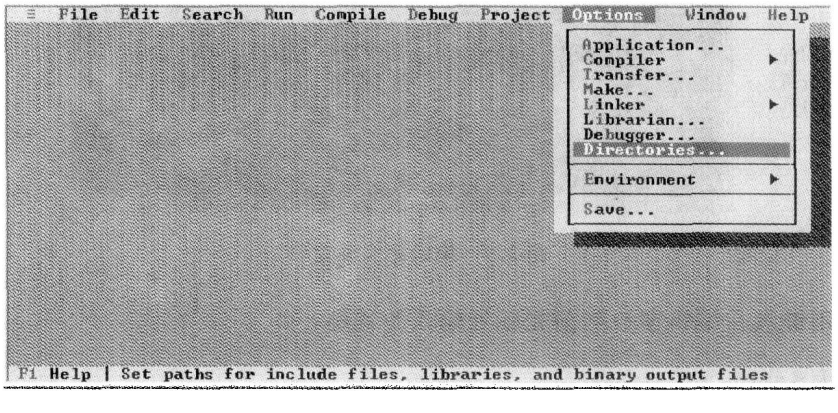


图 2.5 建立工作环境命令菜单

(2) 选中 Directories 命令后，按回车键，将弹出如图 2.6 所示的对话框。

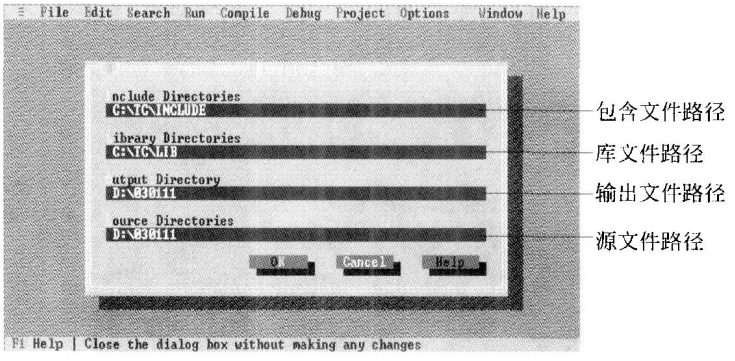


图 2.6 建立工作环境对话框

(3) 在图 2.6 所示对话框可设置包含文件、库文件、输出文件和源文件的路径，其中包含文件和库文件的默认路径，一般不需修改，但若 Turbo C++不是安装在 C 盘下，则要根据实际路径进行修改。Output Directory 文本框中应输入工作目录，例如“D:\030111”，然后按回车键或者单击 OK 按钮即可。**注意：输出文件路径和源文件路径必须存在，否则在编译时会提示找不到输出文件路径和源文件的路径。**

2.2.4 编辑源文件

(1) 在工作窗口中，选中 File 菜单，然后按回车键，将弹出 File 子菜单。

(2) 若需要编辑一个新的文件，则可选中 File 子菜单中的 New 命令，然后按回车键。此时将进入源程序编辑窗口，用户可以在光标闪烁的地方开始编写源程序，如图 2.7 所示。

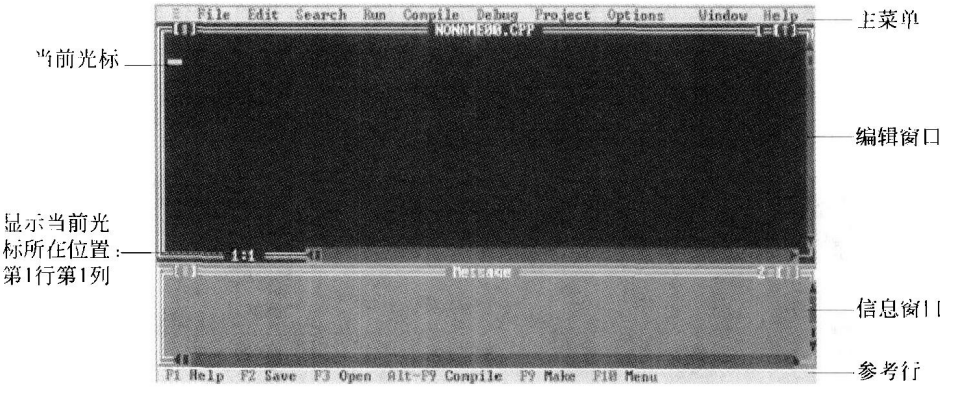


图 2.7 编辑源文件窗口

(3) 使用键盘在编辑窗口当前光标处输入源程序：

```
#include <stdio.h>
void main()
{
```

```
    printf("**************************************\n");
    printf("              Hello,World!\n");
    printf("**************************************\n");
}
```

如果用户需要编辑的文件已经存在，则可选择 File 子菜单中的 Open 命令，按回车键后将弹出一个打开文件的对话框，如图 2.8 所示。直接按 F3 功能键也能打开该对话框。

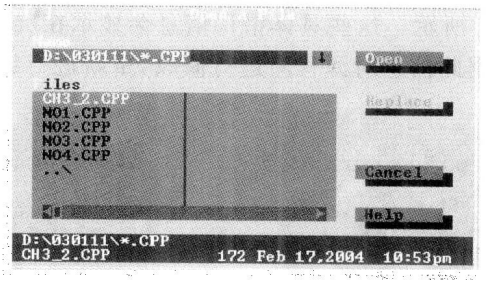


图 2.8 打开文件对话框

使用 Tab 键或光标移动键选中要编辑的已存在源文件，按回车键或单击 Open 按钮源文件的内容将显示在深蓝色的编辑窗口中，用户就可以对其进行编辑和修改了。

2.2.5 保存源文件

源程序在编辑时是保存在计算机内存中的，为了防止意外事故丢失程序，最好将输入的程序存储到磁盘中，也就是进行保存文件的操作。具体方法是选择 File 菜单中的 Save 命令后，再按回车键，或者直接按 F2 键，此时将弹出如图 2.9 所示的文件保存对话框。

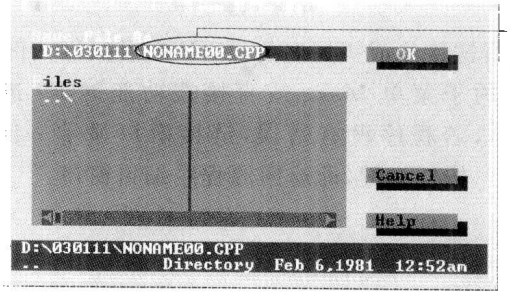


图 2.9 文件保存对话框

如图 2.9 所示，Save File As 文本框中的默认路径为所定义的工作目录 D:\030111，而其默认文件名则为 NONAME00.CPP，在保存时可以把 NONAME00 文件名更改为便于记忆的其他文件名，如 Ch2_1。

2.2.6 编译

程序编辑完后，先编译源代码，生成扩展名为.OBJ 的目标文件。

对单文件程序编译的方法是在文件编辑修改完后，按 F10 键激活主菜单，将亮条移至 Compile 处后回车，通过光标上下移动键选择 Compile 子菜单，或不激活主菜单直接按组合键 Alt＋F9 进行编译，进入编译状态后，屏幕会出现一个编译(Compiling)窗口，几秒钟后，编译窗口显示一闪烁信息：Success: Press any key!，表示编译成功，如图 2.10 所示。此时可按任意键，返回源程序编辑窗口。

如果编译时产生警告 Warning 或出错 Error 信息，编译窗口显示一闪烁信息：Errors: Press any key!，如图 2.11 所示。这些具体错误信息会显示在屏幕下部的 Message 窗口中，如图 2.12 所示。用户根据此信息对源程序进行修改，重新进行编译。

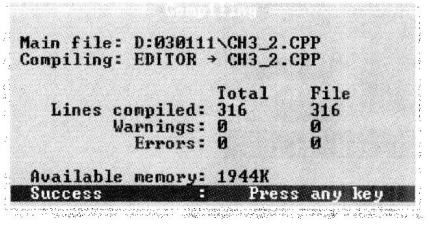


图 2.10 编译成功窗口

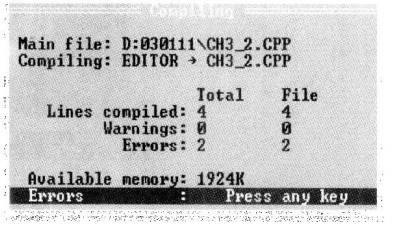


图 2.11 编译失败窗口

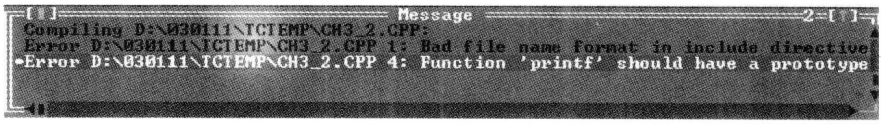


图 2.12 Message 窗口信息

2.2.7 链接

将编译生成的目标文件进行链接生成扩展名为.EXE 的可执行文件。

激活主菜单选择 Compile 的子菜单 Make 或直接按功能键 F9 进行链接，出现链接(Linking)窗口，如图 2.13 所示，若程序没有错误，链接窗口显示一闪烁信息：Success: Press any key! 如图 2.13 所示。按任意键，即返回源程序编辑窗口。

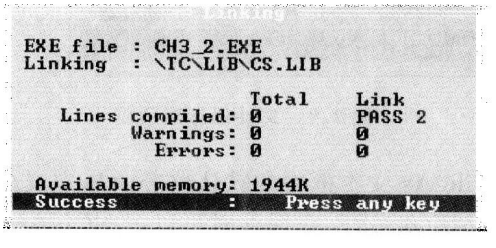


图 2.13 链接成功窗口

若程序有错误，链接窗口显示一闪烁信息：Errors: Press any key! 这些具体错误信息会显示在屏幕下部的 Message 窗口中，用户根据此信息对源程序进行修改，重新进行编译链接。

2.2.8 运行

源程序经编译链接无误后生成一可执行文件，可以投入运行。利用主菜单 Run 中的 Run 命令或直接按 Ctrl+F9 组合键均可运行程序。

其实，调试程序时最简便最常用的方法是在源程序编辑完成后直接用 Run 命令或直接按 Ctrl+F9 组合键。如果源程序没有错误，Turbo C++将一次完成从编译、链接到运行的全过程。若有错误，则会显示编译或链接对话框，提示编译或链接有错误，需要回到 Turbo C 编辑状态根据出错提示信息修改源程序，直至编译链接成功。

程序投入运行时，屏幕会出现一个链接窗口，显示 Turbo C 正在链接和程序所需的库函数。链接完毕后，屏幕会突然一闪，后又回到 Turbo C 主屏幕。这时若想看运行结果，可用主菜单 Window 中的 Use screen 命令或直接按 Alt+F5 转到用户屏幕，此时屏幕被清除，在顶部显示程序运行的结果，如图 2.14 所示。

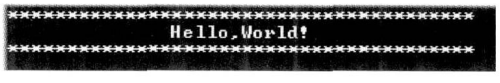


图 2.14 查看程序运行结果

2.2.9 退出 Turbo C

激活主菜单，选择 File 菜单下的子菜单 Quit 命令，或直接通过按组合键 Alt+X，返回操作系统。

至此，一个程序的编辑、编译、链接、运行的全过程就介绍完毕。

2.3 Turbo C++ 3.0 环境下程序的动态调试

编译和链接没有错误不等于运行结果一定正确。编译系统能检查出语法错误，但无法检查出逻辑错误。为此可以在源程序适当位置插入输出语句，输出有关变量的值进行检查。这个方法的缺点是比较麻烦，而且对程序的运行过程无法控制。下面介绍两种动态调试方法。

2.3.1 单步执行方法

单步执行即一次执行一行程序，每执行一行，就暂停下来，人们借此机会可以查看某个变量的值或表达式的计算结果，以便发现问题所在。单步执行可以通过 Run 菜单中的子菜单项实现，或通过功能键实现，如表 2.1 所示。

如果希望查看变量的值或表达式的计算结果，可以通过设置 Watch 窗口来实现，具体的操作如图 2.15 所示。

表 2.1 单步执行调试命令

Run 菜单中的子菜单	功能键	功 能
Trace into	F7	单步执行。令程序执行一个语句，遇到函数调用时执行进入函数体
Step over	F8	与 F7 功能键类似，但执行时不进入函数，把函数调用简单地看作一个语句
Go to cursor	F4	使程序连续执行，直到达到当前光标所在的语句，停在那里

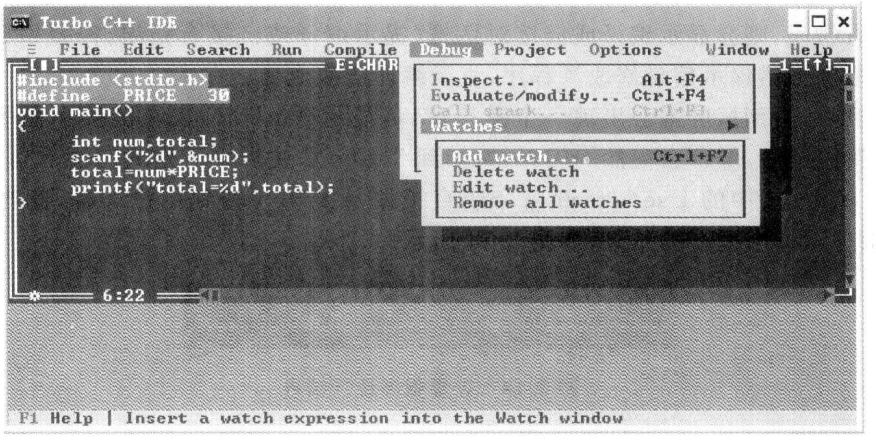


图 2.15 打开“Add Wacth”窗口的命令

首先选中 Debug 菜单，然后选中 Watches 子菜单项中的 Add Watch 命令，打开 Add Watch 窗口，按 Ctrl+F7 也可以打开 Add Watch 窗口。在打开的窗口中输入希望查看的变量名，如图 2.16 所示。

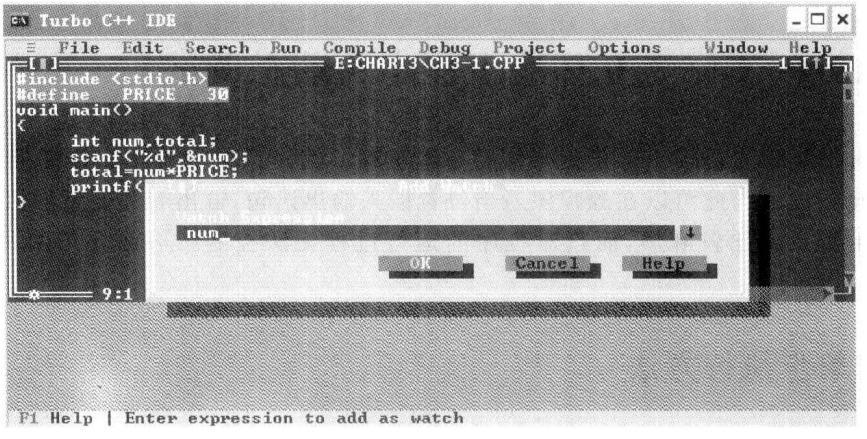


图 2.16 设置观察变量

假设输入 num(一个 Add Watch 窗口只能输入一个观察变量，若要观察两个以上变量，则要多次打开 Add Watch 窗口)，然后单击 OK 按钮，就可以看到如图 2.17 所示的 Watch 窗口中列出的待查看的变量名。由于程序还没有执行，所以它们处于没有定义状态。

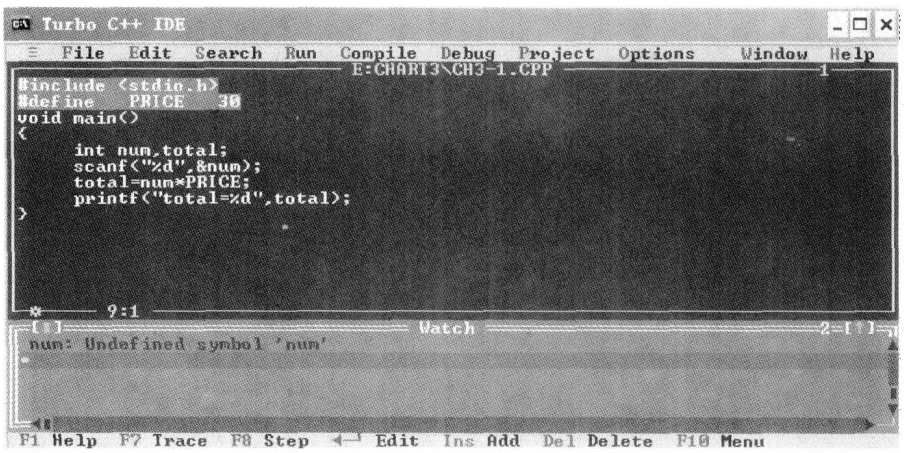


图 2.17 观察变量设置完成

如果想从 Watch 窗口中删去某个观察变量，就使用鼠标选中 Watch 窗口中的相应变量，然后选择 Debug\Watches\Delete watch 命令就可以了。

完成上述设置后，就可以进入单步执行阶段了。按 F7 键后可以看到在编辑窗口中的源程序中的主函数 void main()处用高亮度显示，表示准备进入 main 函数。同时可以看到屏幕下部的 message 窗口变成了 Watch 窗口，它是作为观察数据用的，如图 2.18 所示。

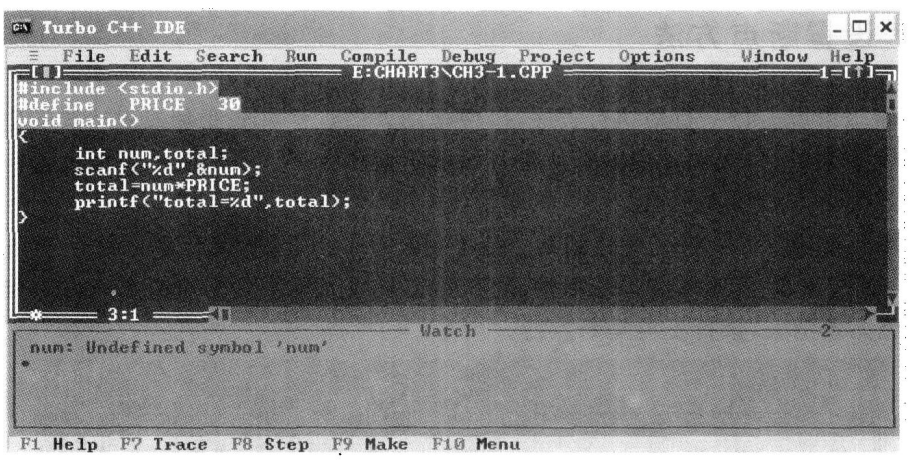


图 2.18 单步执行从 main 函数开始

再按一次 F7 键，亮条移到第 6 行，第 5 行是对变量的定义，不是执行语句，故被跳过。此时已进入 main 函数的左花括号，但并未开始执行第 6 行，只是表明下一步要执行此行。再按一次 F7 键，此时执行第 6 行，由于该行是 scanf 函数语句，需要输入数据，所以切换到用户屏幕，用户在此输入 10 后按回车键，屏幕显示切换到编辑窗口，亮条移到第 7 行，表示第 6 行已执行完毕，观察窗口中 num 的值变为刚输入的 10。再按一次 F7 键，亮条移到第 8 行，表示第 7 行执行完毕。再按一次 F7 键，亮条移到第 9 行右花括号处，表示第 8 行 printf 函数执行完毕，如图 2.19 所示。此时屏幕闪了一下，可以按 Alt+F5 键查看运行结果。

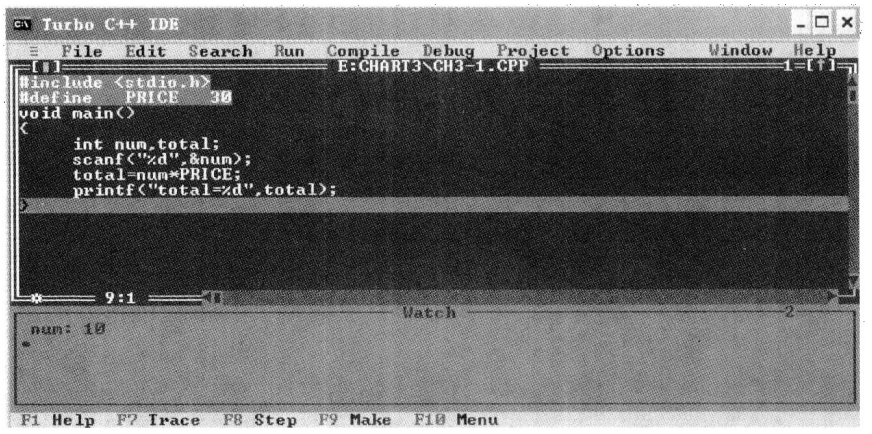


图 2.19 单步执行到 main 函数右花括号处

当亮条移到 main 函数右花括号处时并未表示程序执行完毕，而应再按一次 F7 键，使亮条消失，至此才表示程序执行完毕。

以上通过一个简单的例子详细地介绍了如何用单步执行的方法来进行动态调试。实际上，对于简单的程序，可以很快查出错误所在，不一定需要采用单步执行方法，但对较为复杂的程序单步执行动态调试是很有必要的。

2.3.2 设置断点方法

单步跟踪可以一行一行地执行程序，以便帮助人们观察某些变量或表达式的变化情况。但是如果程序太长，每次调试程序都从头开始就不太现实了。所以我们可以设置一个断点，然后从断点处再开始单步跟踪。

设置断点的方法是：将光标移到某一行上，按 Ctrl+F8 键，此行就以红色亮条显示，作为断点行，如图 2.20 所示。如果想取消断点行，也是将光标移到断点行上，再按一次 Ctrl+F8 键，红色亮条消失，该行不再是断点行。

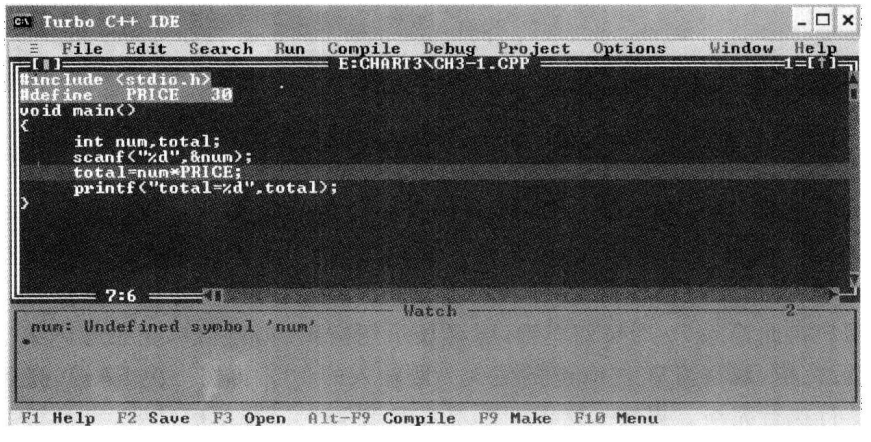


图 2.20 断点设置

运行时遇到断点行将暂停，此时，用户可以用单步执行方法查看有关变量或表达式的值。如果想继续运行，再按一次 Ctrl+F9 键即可。

断点设置也可以通过菜单中 Debug\toggle breakpoint 命令完成，但用功能键比菜单选择方便得多。

以上只是初步介绍了使用调试工具调试程序的方法。在此基础上，读者可以进一步掌握系统提供的其他调试工具，以便更有效地进行调试工作。

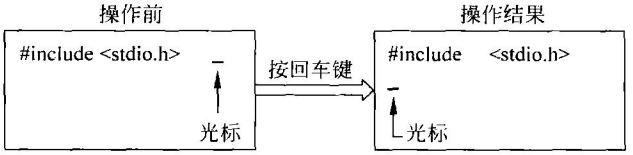
2.4 常用编辑键和菜单命令热键

在应用程序开发环境中，要求会使用一些常用的编辑键、快捷菜单和功能键的操作方法，以提高上机调试程序的效率。

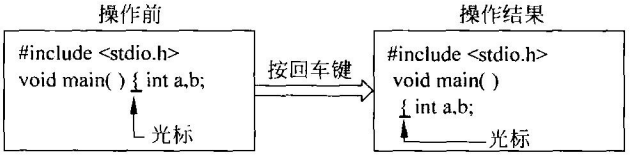
2.4.1 常用编辑键

1. 回车换行的 Enter 键

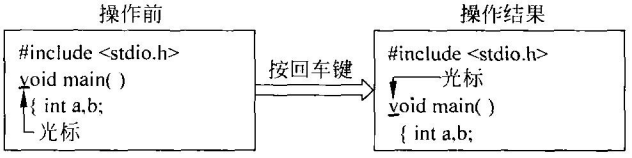
【例 1】 输入完一行后，光标转到下一行。



【例 2】 将一行分为两行。



【例 3】 在两行之间加一空行。



2. 将光标移到行尾的 End 键